37

Pose Estimation of Cycloidal Curves by using Twist Representations

Bodo Rosenhahn and Gerald Sommer

ABSTRACT This work concerns the 2D-3D pose estimation problem of cycloidal curves. Pose estimation means to estimate the relative position and orientation of a 3D object to a reference camera system. The 3D object features are in this work cycloidal curves, as extensions to classical 3D point or 3D line concepts. This means, we assume knowledge of a 3D cycloidal curve and observe it in an image of a calibrated camera. The aim is to estimate the rotation $\mathbf{R}$ and translation $\mathbf{t}$ to get a best fit of the transformed 3D object model to the observed 2D image data. Furthermore, other concepts such as 3D cycloidal surfaces and the numerical problems of estimating the pose parameters are discussed.

Keywords: 2D-3D pose estimation, cycloidal curves, twists

37.1 Introduction

In this paper we study the 2D-3D pose estimation problem of 3D cycloidal curves. Cycloidal curves are a special case of algebraic curves. In general a cycloidal curve is generated by a circle rolling on a circle or line without slipping [11]. This leads to epitrochoids, hypotrochoids and trochoids as special classes of entities, we will use in the context of 2D-3D pose estimation.

37.1.1 The pose estimation problem

Pose estimation itself is a basic visual task [6] and several approaches for monocular pose estimation exist that relate the position of a 3D object to a reference camera coordinate system [6, 9, 19]. Our preliminary work also concerned the 2D-3D pose estimation problem of rigid objects and kinematic chains [16]. Instead of using invariance as an explicit formulation of geometry, as often has been done in projective geometry, we are using implicit formulations and use constraints to describe the pose estimation problem. The formulas in [15] result in compact constraint equations for pose estimation of rigid objects containing different entities (points, lines, planes). The entities we are now interested in are cycloidal curves.

AMS Subject Classification: 65D17, 68U10.

This means curves we can generate by one, two or more coupled twists. Examples are 3D circles, ellipses, cardioids, cycloids, archimedic spirals, spheres, quadrics, etc. In this paper we will use conformal geometric algebra (CGA) [10] to describe schemes for cycloidal curves and their coupling with the 2D-3D pose estimation problem.

37.1.2 Preliminary work

Our recent work [15, 16] is summarized in Figure 37.1: There we assume as object features 3D points, 3D lines, 3D spheres, 3D circles or kinematic chain segments of a reference model. Further, we assume corresponding extracted features in an image of a calibrated camera. The aim is to find the rotation $\mathbf{R}$ and translation $\mathbf{t}$ of the object, which leads to the best fit of the reference model with the actual extracted entities. To relate 2D image information to 3D entities we reconstruct an extracted image entity to an entity with one dimension higher gained by back-projection in the space. This idea will be used to formulate the problem as a pure kinematic problem.

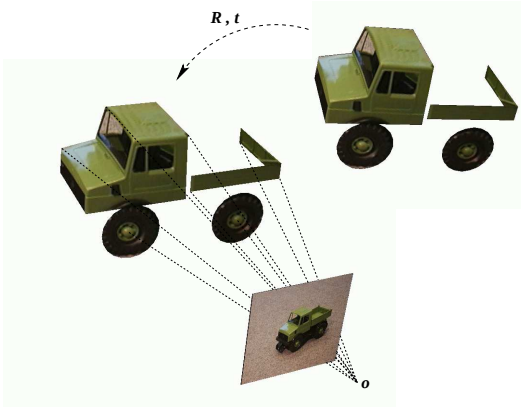

FIGURE 37.1. The scenario. The assumptions are the projective camera model, the model of the object (consisting of points, lines, circles and kinematic chains) and corresponding extracted entities on the image plane. The dashed line describes the pose of the model, which leads to the best fit of the object with the actual extracted entities.

During the last few years we studied several algebras (e.g., the algebras for pure projective [14] or pure kinematic geometry [1]), but the conformal geometric algebra is the most general one that describes the problems involved in pose estimation. The mathematical spaces involved in the pose estimation problem are the projective plane, projective space, kinematic space and Euclidean space. Indeed all geometric algebras for modeling the spaces involved in the pose estimation problem are represented as sub-algebras of the conformal geometric algebra. So we are able to formalize all aspects of the pose estimation problem in one framework.

The motivation for this work is to generalize the already studied entities for pose estimation to more *free-form*-like contours and surfaces. Though points, lines, planes, circles and spheres cover a large range to model objects, we are also interested in, e.g., ellipses to model the shape of eyes etc. This motivates a search for a more general class of entities, which can be used in the context of pose estimation. We now give a very brief summary of algebraic curves and their characterizations collected by Lee in [11]. More detailed information about algebraic curves can also be found in [4]. In our work we will concentrate on a

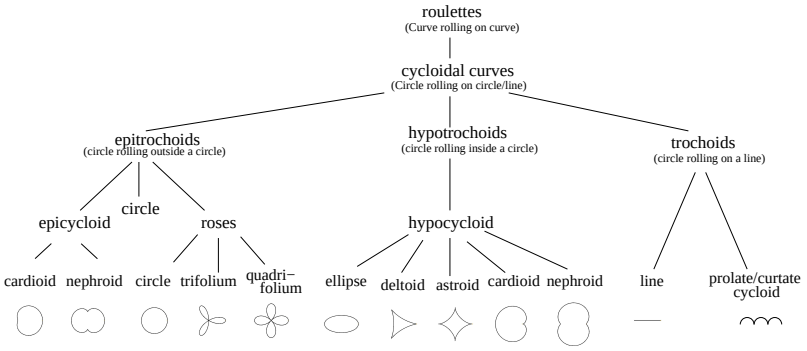

FIGURE 37.2. Tree of algebraic curves.

subclass of *roulettes*, the *cycloidal curves*, which are circles rolling on circles or lines. Figure 37.2 shows a subtree of the family of algebraic curves. Cycloidal curves can be classified as epitrochoids, hypotrochoids and trochoids, which split into further subclasses. Figure 37.2 also shows examples of these curves.

These curves are mostly defined in the 2D plane. For our scenario of pose estimation, we will extend these curves to plane curves in the 3D space.

37.2 Cycloidal curves in conformal geometric algebra

In this section we will introduce the conformal geometric algebra with respect to the basic notation and the description of cycloidal curves in the language of parameterized twist transformations.

37.2.1 Introduction to conformal geometric algebra

In this section we introduce the main properties of the conformal geometric algebra [10]. The aim is to clarify the notation, a more detailed introduction concerning geometric algebras can be found in [18].

A geometric algebra $\mathcal{G}_{p,q,r}$ is built from a vector space $\mathbb{R}^n$, endowed with the signature (p, q, r), $n = p + q + r$, by application of a geometric product. The product defining a geometric algebra is called *geometric product* and is denoted

by juxtaposition, e.g., $\boldsymbol{uv}$ for two multivectors $\boldsymbol{u}$ and $\boldsymbol{v}$. The geometric product consists of an outer $(\wedge)$ and an inner $(\cdot)$ product, whose roles are to increase or to decrease the order of the algebraic entities, respectively. For later use we introduce the commutator $\underline{\times}$ and anticommutator $\overline{\times}$ products, respectively for any two multivectors,

$$\boldsymbol{AB} = \tfrac{1}{2}(\boldsymbol{AB} + \boldsymbol{BA}) + \tfrac{1}{2}(\boldsymbol{AB} - \boldsymbol{BA}) =: \boldsymbol{A}\overline{\times}\boldsymbol{B} + \boldsymbol{A}\underline{\times}\boldsymbol{B}.$$

To introduce the conformal geometric algebra (CGA) we follow [10] and start with the *Minkowski plane* $\mathbb{R}^{1,1}$, which has an orthonormal basis $\{\mathbf{e}_+, \mathbf{e}_-\}$, defined by the properties

$$\mathbf{e}_+^2 = 1, \quad \mathbf{e}_-^2 = -1, \quad \text{and} \quad \mathbf{e}_+ \cdot \mathbf{e}_- = 0.$$

A *null basis* can now be introduced by the vectors

$$\mathbf{e}_0 := \tfrac{1}{2}(\mathbf{e}_- - \mathbf{e}_+) \quad \text{and} \quad \mathbf{e} := \mathbf{e}_- + \mathbf{e}_+$$

with $\mathbf{e}_0^2 = \mathbf{e}^2 = 0$. The vector $\mathbf{e}_0$ can be interpreted as the origin and the vector $\mathbf{e}$ as a point at infinity. Furthermore we define $\boldsymbol{E} := \mathbf{e} \wedge \mathbf{e}_0 = \mathbf{e}_+ \wedge \mathbf{e}_-$.

In the case of working in an n-dimensional vector space $\mathbb{R}^n$ we couple an additional vector space $\mathbb{R}^{1,1}$, which defines a null space to gain $\mathbb{R}^n \oplus \mathbb{R}^{1,1} = \mathbb{R}^{n+1,1}$. From that vector space we can derive the conformal geometric algebra (CGA) $\boldsymbol{\mathcal{G}}_{n+1,1}$ as a linear space of dimension 2^{n+2}.

The algebras $\boldsymbol{\mathcal{G}}_{3,1}$ and $\boldsymbol{\mathcal{G}}_{3,0}$ are suited to represent the projective and Euclidean space, respectively [8]. Since

$$\boldsymbol{\mathcal{G}}_{4,1} \supseteq \boldsymbol{\mathcal{G}}_{3,1} \supseteq \boldsymbol{\mathcal{G}}_{3,0},$$

both algebras for the projective and Euclidean space constitute subspaces of the linear space of the CGA. It is possible to use operators to relate the different algebras and to guarantee the mapping between the algebraic properties, see [17].

The basis entities of the 3D conformal space are spheres $\underline{\boldsymbol{s}}$, containing the center $\boldsymbol{p}$ and the radius ρ, $\underline{\boldsymbol{s}} = \boldsymbol{p} + \frac{1}{2}(\boldsymbol{p}^2 - \rho^2)\mathbf{e} + \mathbf{e}_0$. Thus, a sphere is a 1-blade. The dual form for a sphere is $\underline{\boldsymbol{s}}^\star$. The advantage of the dual form is that $\underline{\boldsymbol{s}}^\star$ can be calculated directly from points on the sphere: For four points on the sphere, $\underline{\boldsymbol{s}}^\star$ can be written as $\underline{\boldsymbol{s}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}} \wedge \underline{\boldsymbol{d}}$ and is therefore a 4-blade.

A point $\underline{\boldsymbol{x}} = \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^2\mathbf{e} + \mathbf{e}_0$ is nothing more than a degenerate sphere with radius $\rho = 0$, which can easily be seen from the representation of a sphere. A point $\underline{\boldsymbol{x}}$ lies on a sphere $\underline{\boldsymbol{s}}$ iff $\underline{\boldsymbol{x}} \wedge \underline{\boldsymbol{s}}^\star = 0$, or $\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{s}} = 0$.

Circles $\underline{\boldsymbol{z}}$ can be described by the intersection of two spheres. The dual representation of circles leads to the outer product of three points on the circle, $\underline{\boldsymbol{z}}^\star = \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}$.

From the dual form of circles and spheres, affine points $\underline{\boldsymbol{X}}^\star$, lines $\underline{\boldsymbol{L}}^\star$ and planes $\underline{\boldsymbol{P}}^\star$ can easily be represented as degenerate cases, just by involving the point at infinity,

$$\underline{\boldsymbol{X}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{x}}, \qquad \underline{\boldsymbol{L}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}}, \qquad \underline{\boldsymbol{P}}^\star = \mathbf{e} \wedge \underline{\boldsymbol{a}} \wedge \underline{\boldsymbol{b}} \wedge \underline{\boldsymbol{c}}.$$

Note that since we will only work with the entities in their dual representation in the next sections, we neglect the $\star$-sign in the further formulas.

37.2.2 Rigid motions and twists in CGA

As in $\mathcal{G}_{3,0}$, rotations in $\mathcal{G}_{4,1}$ are represented by rotors $\mathbf{R} = \exp\left(-\frac{\theta}{2}\boldsymbol{l}\right)$. The components of the rotor $\mathbf{R}$ are the unit bivector $\boldsymbol{l}$, which represents the dual of the rotation axis and the angle θ, which represents the amount of the rotation. If we want to translate an entity with respect to a translation vector $\mathbf{t} \in \mathcal{G}_{3,0}$, we can use a so-called *translator*, $\boldsymbol{T} = (1 + \frac{\mathbf{e_t}}{2}) = \exp\left(\frac{\mathbf{e_t}}{2}\right)$, which is a special rotor. Rotations and translations can be estimated by applying rotors and translators as *versor products*, e.g., $\underline{\boldsymbol{X}}' = \mathbf{R}\underline{\boldsymbol{X}}\widetilde{\mathbf{R}}$, or $\underline{\boldsymbol{X}}'' = \boldsymbol{T}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{T}}$. Note, that we use the $\sim$-sign on a multivector to denote its reverse. To express a rigid body motion, we can apply multiplied rotors and translators consecutively. We denote such an operator as a motor $\boldsymbol{M}$ [1]. The rigid body motion of, e.g., a point $\underline{\boldsymbol{X}}$ can be written as $\underline{\boldsymbol{X}}' = \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}$, see also [7].

Following e.g., [12], a rigid body motion of points can be expressed by a rotation around a line in space followed by a translation along this line. This results from the fact, that for every $g \in SE(3)$ there exists a $\xi \in se(3)$ and a $\theta \in \mathbb{R}$ such that $g = \exp(\xi\theta)$. Such transformations are also called *twist* transformations. The Lie algebra element $\xi \in se(3)$ is a twist, and its Lie group element, the exponential $g = \exp(\xi\theta) \in SE(3)$ describes a rigid body motion [5]. A motor describing a twist transformation can be written as

$$\boldsymbol{M} = \exp\left(-\tfrac{1}{2}\theta\left(\boldsymbol{l} + \mathbf{e}\boldsymbol{m}\right)\right) = \exp\left(-\tfrac{1}{2}\theta\Psi\right).$$

A twist can be seen as an infinitesimal version of a screw motion and describes a line in space with an angle θ and a *pitch* h, the ratio of translation to rotation. If the pitch h is zero, the resulting motion is a rotation of an entity (e.g., a point $\underline{\boldsymbol{X}}$) around a line $\underline{\boldsymbol{L}}^{\star}$ in the space. To gain a twist representation, the general idea is to translate both, the entity and the line to the origin, to perform a rotation and to translate back the transformed entity.

The motor $\boldsymbol{M}$ can be interpreted as the exponential of a twist, with the form

$$\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}\widetilde{\boldsymbol{T}} = \exp\left(-\tfrac{1}{2}\theta\left(\boldsymbol{l} + \mathbf{e}(\boldsymbol{t}\cdot\boldsymbol{l})\right)\right).$$

The motion of a point can then be decomposed as

$$\underline{\boldsymbol{X}}' = \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}} = (\boldsymbol{T}\boldsymbol{R}\widetilde{\boldsymbol{T}})\underline{\boldsymbol{X}}(\boldsymbol{T}\widetilde{\boldsymbol{R}}\widetilde{\boldsymbol{T}}).$$

We call such a transformation a *general rotation*. Whereas in Euclidean geometry, Lie algebras and Lie groups are only applied to point concepts, motors can also be applied to other entities, like lines, planes, circles, spheres, etc.

37.2.3 Cycloidal curves in conformal geometric algebra

As previously explained, cycloidal curves are circles rolling on circles or lines. In this section we will explain how to generate such curves in conformal geometric

algebra. E.g., ellipses are not entities which can be directly described in conformal

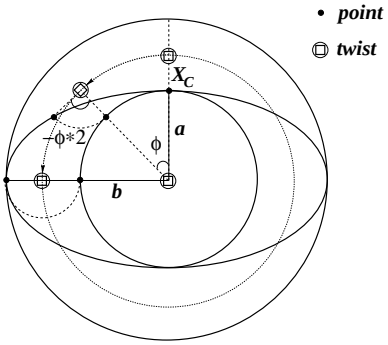

FIGURE 37.3. An ellipse generated by two coupled twists.

geometric algebra. The idea for modeling an ellipse is visualized in Figure 37.3: We assume two parallel twists (modeling general rotations) in the 3D space and a 3D point on the ellipse, and we transform the point around the two twists in a fixed and dependent manner. In this case, we use two coupled parallel (not collinear) twists, rotate the point by -2ϕ around the first twist and by ϕ around the second one. The set of all points for $\phi \in [0, \ldots, 2\pi]$ generates an ellipse as the orbit of the coupled group actions. In general, every cycloidal curve is generated by a

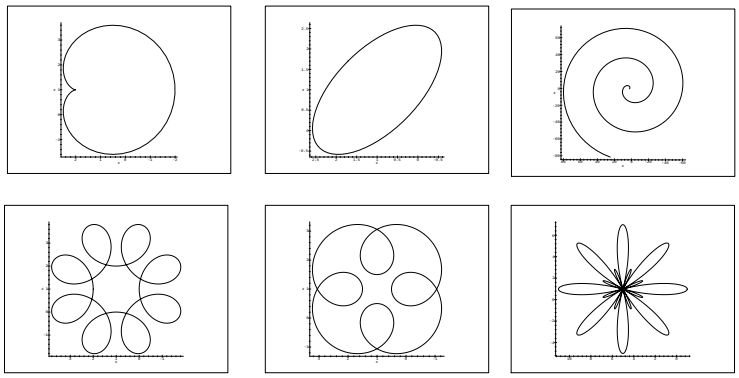

FIGURE 37.4. 3D-2twist generated curves.

set of twists ξ_i with frequencies λ_i acting on one point $\underline{\boldsymbol{X}}$ on the curve. Since twists can be used to describe general rotations in the 2D plane or 3D space, we call the generated curves *n*D-*m*twist curves. With *n*D-*m*twist curves we mean n dimensional curves, generated by m twists with $n, m \in \mathbb{N}$. In the context of the 2D-3D pose estimation problem we use the cycloidal curves as 3D object entities. So we mean 3D-$m$twist curves, if we speak of just $m$twist curves.

We will start with very simple curves, and the simplest one consists of one point (a point on the curve) and one twist. Rotating the point around the twist leads to the parameterized generation of a circle: The corresponding twist transformation can be expressed as a suitable motor $\boldsymbol{M}_\phi$ and an arbitrary 3D point, $\underline{\boldsymbol{X}}_Z$, on the circle. The 3D orbit of all points on the circle is simply given by

$$\underline{\boldsymbol{X}}_Z^\phi = \boldsymbol{M}_\phi \underline{\boldsymbol{X}}_Z \widetilde{\boldsymbol{M}}_\phi, \qquad \phi \in [0, \ldots, 2\pi].$$

We call a circle also a *1twist* generated curve since it is generated by one twist.

Now we can continue and wrap a second twist around the first one. If we make the amount of rotation of each twist dependent on each other, we gain a 3D curve in general. This curve is firstly dependent on the relative positions and orientation of the twists with respect to each other, the (starting) point on the curve and the ratio of angular frequencies.

The general form of a *2twist* generated curve is

$$\underline{\boldsymbol{X}}_C^\phi = \boldsymbol{M}^2_{\lambda_2\phi} \boldsymbol{M}^1_{\lambda_1\phi} \underline{\boldsymbol{X}}_C \widetilde{\boldsymbol{M}}^1_{\lambda_1\phi} \widetilde{\boldsymbol{M}}^2_{\lambda_2\phi}, \quad \lambda_1, \lambda_2 \in \mathbb{R}, \;\; \phi \in [\alpha_1, \ldots, \alpha_2].$$

The motors $\boldsymbol{M}^i$ are the exponentials of twists, the scalars $\lambda_i \in \mathbb{R}$ determine the ratio of angular frequencies between the twists and $\underline{\boldsymbol{X}}_C$ is a point on the curve. The values α_i define an interval for the boundaries of the curve. For closed curves they are usually $\alpha_1 = 0$ and $\alpha_1 = 2\pi$, but indeed it is also possible to define curve segments. The case $\lambda_1 = \lambda_2 = 1$ leads to cardioids, $\lambda_1 = 2$, $\lambda_2 = 1$ leads

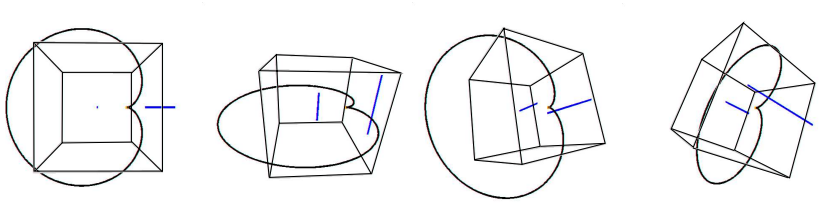

FIGURE 37.5. Perspective views of a 3D-2twist generated curve. The 2twist curve and the twists axes are visualized.

to nephroids and $\lambda_1 = 3, \lambda_2 = 1$ leads to deltoids, which can be transformed (by moving the second twist) to a trifolium, etc. Ellipses can easily be described by $\lambda_1 = -2, \lambda_2 = 1$.

Figure 37.4 shows further examples from curves, which can be very easily generated by two coupled twists. Note: Also the archimedic spiral is a 2twist generated curve. To gain an archimedic spiral, one twist has to be a translator. Since an archimedic spiral is no algebraic curve, the 2twist generated curves are more general than the cycloidal curves.

All these curves are given in a 3D space. In Figure 37.4 only projections are shown. Figure 37.5 shows a 3D 2twist generated curve of different projective views.

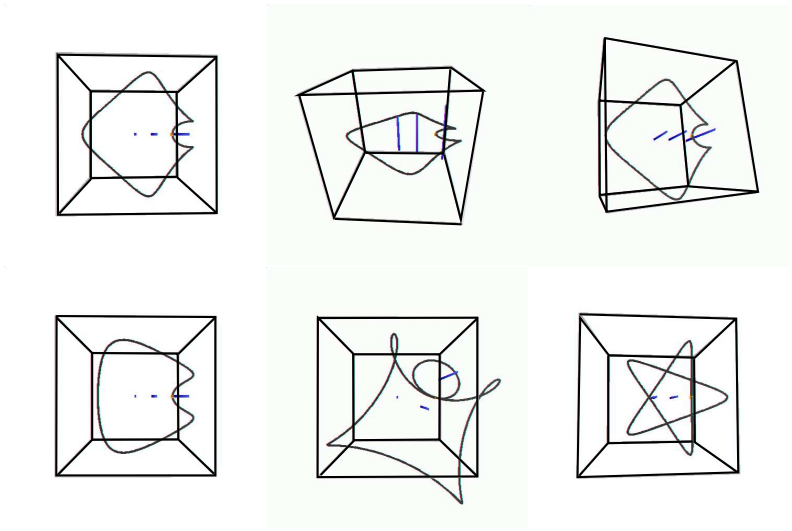

FIGURE 37.6. 3D-3twist generated curves. The first row shows three different perspective views of a 3twist generated curve. The other row shows other 3twist generated curves, generated with different frequencies and twist positions. The three twists are also visualized.

It is also possible to generate 3twist curves. For this, a point is transformed with respect to three twists. Examples of planar curves are shown in Figure 37.6.

So far, we have only formalized 3D curves. Surfaces can be modeled by rotating, e.g., the 2twist generated curves around a third twist with a second independent angle ϕ_2, leading to 3twist generated surfaces. Examples are shown in Figure 37.7. Note: If there is only one variable angle ϕ, the resulting entity is always a 3D curve in the 3D space, whereas the case of two variable angles ϕ_1 and ϕ_2 leads to a 3D surface in the 3D space. The case of three variable angles leads to volumes as highest structures in the 3D space.

The general form of 3twist generated surfaces is

$$\underline{\boldsymbol{X}}^{\phi_1,\phi_2} = \boldsymbol{M}^3_{\lambda_3\phi_2}\boldsymbol{M}^2_{\lambda_2\phi_1}\boldsymbol{M}^1_{\lambda_1\phi_1}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}^1_{\lambda_1\phi_1}\widetilde{\boldsymbol{M}}^2_{\lambda_2\phi_1}\widetilde{\boldsymbol{M}}^3_{\lambda_3\phi_2},$$

with $\lambda_i \in \mathbb{R}$ and $\phi_1, \phi_2 \in [\alpha_{i_1}, \ldots, \alpha_{i_2}]$. An ellipsoid, for example, is nothing more than a (special) rotated ellipse ($\lambda_3 = \lambda_2 = 1, \lambda_1 = -2$). Its parameterized equation can be written as

$$\underline{\boldsymbol{X}}^{\phi_1,\phi_2}_Q = \boldsymbol{M}^3_{\phi_2}\boldsymbol{M}^2_{\phi_1}\boldsymbol{M}^1_{-2\phi_1}\underline{\boldsymbol{X}}_Q\widetilde{\boldsymbol{M}}^1_{-2\phi_1}\widetilde{\boldsymbol{M}}^2_{\phi_1}\widetilde{\boldsymbol{M}}^3_{\phi_2}, \quad \phi_1, \phi_2 \in [0, \ldots, 2\pi].$$

This is visualized in the first image of Figure 37.7. The second image of Figure 37.7 shows a horizontally rotated cardioid.

The third and fourth images in the second row show rotated hypocycloids. The last row shows rotated spirals, leading to surfaces, comparable to a flower.These surfaces are very easy to generate and can be represented by just a few coupled twists. In Figure 37.7 no grid-plot is shown. Instead the rotated cycloidal curves

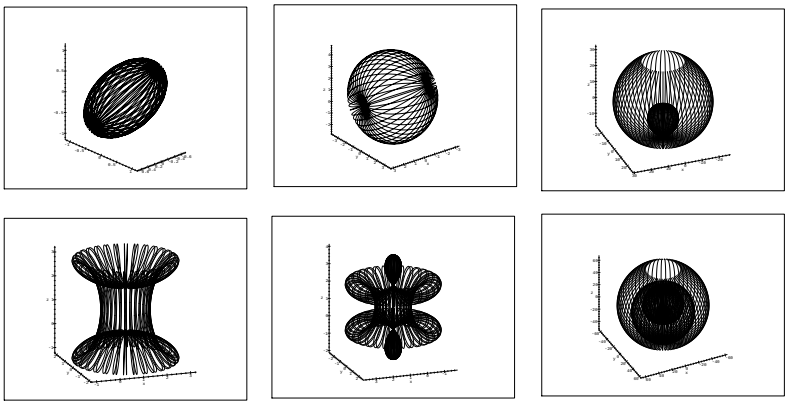

FIGURE 37.7. 3D-3twist generated surfaces

are shown to visualize the geometric generation of the surface. Lines and planes also fall into the definition of 2twist and 3twist generated curves and surfaces. Examples for well-known entities, interpreted as cycloidal curves or surfaces, are points, lines, circles, ellipses, line segments, spirals, spheres, ellipsoids, planes, cylinders, cones, etc.

The rigid body motions of these entities can easily be estimated, just by transforming the generating twists. The transformation of an m twist generated curve can be performed by transforming the m twists (which are just lines in the space), and the point on the curve.

The description of these curves is compact, and rigid transformations can be estimated very quickly. Cycloidal curves and surfaces extend already studied entities to a more general class of entities without losing the advantages of our previous work. Indeed, we build up a hierarchy of entities and further work will concentrate on enlarging these kinds of entities to approximate free-form curves and surfaces. So far the hierarchy of entities consists of the following entities:

points, lines	$\subseteq$	circles	$\subseteq$	cycloidal curves	$\subseteq \ldots$
planes	$\subseteq$	spheres	$\subseteq$	cycloidal surfaces	$\subseteq \ldots$

37.3 Pose estimation in conformal geometric algebra

Now we will formalize the 2D-3D pose estimation problem, that is, *a transformed object entity must lie on a projective reconstructed image entity*. Let $\underline{\boldsymbol{X}}$ be an object point and $\underline{\boldsymbol{L}}$ be an object line, given in CGA. The (unknown) transformation of the entities can be written as $\boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}$ and $\boldsymbol{M}\underline{\boldsymbol{L}}\widetilde{\boldsymbol{M}}$.

Let $\boldsymbol{x}$ be an image point and $\boldsymbol{l}$ be an image line on a projective plane. The

projective reconstruction from an image point in CGA can be written as $\underline{\boldsymbol{L}}_x = \mathbf{e} \wedge \boldsymbol{O} \wedge \boldsymbol{x}$. This leads to a reconstructed projection ray, containing the optical center $\boldsymbol{O}$ of the camera, see Figure 37.1, the image point $\boldsymbol{x}$ and the vector $\mathbf{e}$ as the point at infinity. Similarly, $\underline{\boldsymbol{P}}_l = \mathbf{e} \wedge \boldsymbol{O} \wedge \boldsymbol{l}$ leads to a reconstructed projection plane in CGA.

Collinearity and coplanarity can be expressed by the commutator and anticommutator products, respectively. Thus, the constraint equations of pose estimation from image points read

$$\underbrace{\underbrace{(\boldsymbol{M} \quad \underbrace{\underline{\boldsymbol{X}}}_{\text{object point}} \quad \widetilde{\boldsymbol{M}})}_{\text{rigid motion of the object point}} \quad \underline{\times} \quad \underbrace{\mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})}_{\substack{\text{projection ray,} \\ \text{reconstructed from the image point}}}}_{\substack{\text{collinearity of the transformed object} \\ \text{point with the reconstructed line}}} = 0,$$

Constraint equations to relate 2D image lines to 3D object points, or 2D image lines to 3D object lines, can also be expressed in a similar manner.

The constraint equations implicitly represent a distance measure which has to be zero. Such compact equations subsume the pose estimation problem at hand: find the best motor $\boldsymbol{M}$ which satisfies the constraint.

37.3.1 The pose estimation problem of cycloidal curves

Now we can combine the last two subsections to formalize the pose estimation problem for cycloidal curves: We assume a 3D cycloidal curve, e.g.,

$$\underline{\boldsymbol{X}}_Z^{\phi} = \boldsymbol{M}^2_{\lambda_1\phi} \boldsymbol{M}^1_{\lambda_2\phi} \underline{\boldsymbol{X}} \widetilde{\boldsymbol{M}}^1_{\lambda_2\phi} \widetilde{\boldsymbol{M}}^2_{\lambda_1\phi}, \quad \lambda_1, \lambda_2 \in \mathbb{R}, \;\; \phi \in [0, \ldots, 2\pi].$$

The rigid motion of this curve incident on a projection ray can be expressed as

$$\left(\boldsymbol{M} (\boldsymbol{M}^2_{\lambda_1\phi} \boldsymbol{M}^1_{\lambda_2\phi} \underline{\boldsymbol{X}} \widetilde{\boldsymbol{M}}^1_{\lambda_2\phi} \widetilde{\boldsymbol{M}}^2_{\lambda_1\phi}) \widetilde{\boldsymbol{M}} \right) \underline{\times} (\mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})) = 0.$$

Since every aspect of the 2D-3D pose estimation problem of cycloidal curves is formalized in CGA, the constraint equation describing the pose problem is compact and easy to interpret: The inner parenthesis contains the parameterized representation of the cycloidal curve with one unknown angle ϕ. The outer parenthesis contains the unknown motor $\boldsymbol{M}$, describing the rigid body motion of the 3D cycloidal curve. This is the pose we are interested in. The expression is then combined, via the commutator product, with the reconstructed projection ray and has to be zero. This describes the incidence of the transformed curve to a projection ray. The unknowns are the six parameters of the rigid motion $\boldsymbol{M}$ and the angle ϕ for each point correspondence.

37.3.2 The pose estimation problem of cycloidal surfaces

Similar to the previous section, it is possible to formalize constraint equations for incidence of cycloidal surfaces to projection rays,

$$\left(\boldsymbol{M}(\boldsymbol{M}^3_{\lambda_3\phi_2}\boldsymbol{M}^2_{\lambda_2\phi_1}\boldsymbol{M}^1_{\lambda_1\phi_1}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}^1_{\lambda_1\phi_1}\widetilde{\boldsymbol{M}}^2_{\lambda_2\phi_1}\widetilde{\boldsymbol{M}}^3_{\lambda_3\phi_2})\widetilde{\boldsymbol{M}}\right) \underline{\times} (\mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})) = 0.$$

But this would not cover all geometric aspects of the surface in the pose problem. It is more efficient to build constraints on the surface contour in the image and to model also tangentiality within the constraints. Therefore we use the surface tangential plane $\underline{\boldsymbol{P}_{\boldsymbol{x}}}$ at each point $\underline{\boldsymbol{X}}$ and claim incidence of the tangential plane $\underline{\boldsymbol{P}_{\boldsymbol{x}}}$ with each projection ray,

$$\left(\boldsymbol{M}(\boldsymbol{M}^3_{\lambda_3\phi_2}\boldsymbol{M}^2_{\lambda_2\phi_1}\boldsymbol{M}^1_{\lambda_1\phi_1}\underline{\boldsymbol{P}_{\boldsymbol{x}}}\widetilde{\boldsymbol{M}}^1_{\lambda_1\phi_1}\widetilde{\boldsymbol{M}}^2_{\lambda_2\phi_1}\widetilde{\boldsymbol{M}}^3_{\lambda_3\phi_2})\widetilde{\boldsymbol{M}}\right) \overline{\times} (\mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})) = 0.$$

The unknowns of these constraint equations are the rigid motion $\boldsymbol{M}$ and the angles ϕ_1 and ϕ_2.

37.4 Experiments

This section shows experimental results.

37.4.1 Estimation of twist parameters

To solve the constraint equations for simple objects, partially represented by cycloidal curves, we linearize the equations with respect to the motors and use the first order Taylor series expression for approximation.

This leads to a mapping of the above mentioned global transformation to a twist representation, which enables incremental changes of pose. This means, we do not search for the parameters of the Lie group $SE(3)$ to describe the rigid body motion [5], but for the parameters which generate their Lie algebra $se(3)$. This idea is taken from [12]. From the resulting linear equations in the unknown 3D rigid body motion we get an error function to be minimized. The main problem of pose estimation of cycloidal curves is that they are in general not convex. This results in the problem of getting trapped in local minima. To avoid global minima, in our first approach we use local search [2] strategies to handle these problems and to find a global minimum.

37.4.2 Pose estimation of convex objects

We will continue with pose estimation results of convex objects. For these examples the gradient method converges directly. The pose, for these types of curves,

can be estimated in nearly video real-time (10 frames per second) on a SUN Ultra 10.

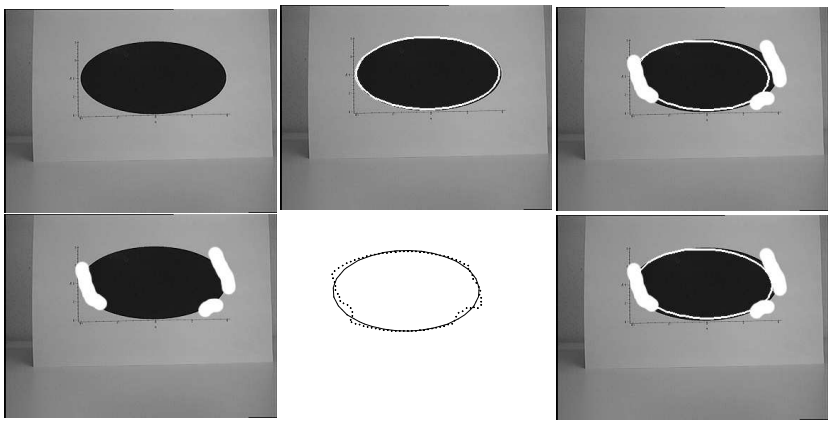

FIGURE 37.8. Pose estimation of a 3D ellipse by using undistorted, distorted and interpolated data.

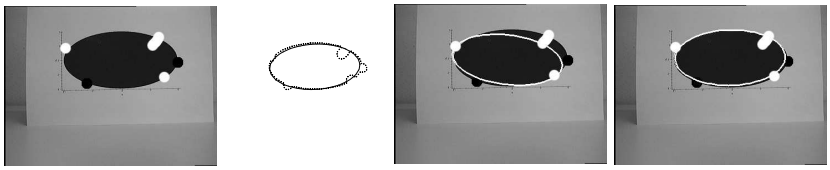

FIGURE 37.9. Pose estimation of a 3D ellipse by using distorted and interpolated data.

Figures 37.8 and 37.9 address the self regularizing properties of image contours: In the first column of Figure 37.8 an undistorted and a distorted image ellipse are shown. The second column shows a pose estimation result found by taking the undistorted image data and the extracted contour of the distorted image. There are two possibilities to deal with the contour points:

On the one hand it is possible to use them directly, on the other hand it is possible to interpolate them to an image ellipse and use the interpolated data. In [20] several approaches for ellipse fitting are presented. To interpolate our contour points, we re-implemented and used the LLS (linear least squares) approach in our scenario. Though the LLS approach is not the best algorithm discussed in [20], it is easy to implement and very fast. The pose results for the raw contour points on the one hand, and the interpolated ellipse points on the other hand, are shown in the third column of Figure 37.8. The upper image shows the pose result achieved by using the pure points, the lower image shows the result achieved by using the interpolated data. Indeed, using the pure points leads to worse results in

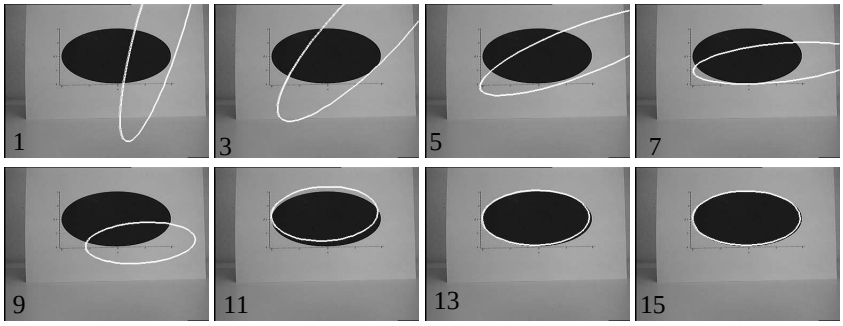

FIGURE 37.10. Convergence behavior of the algorithm during the iteration.

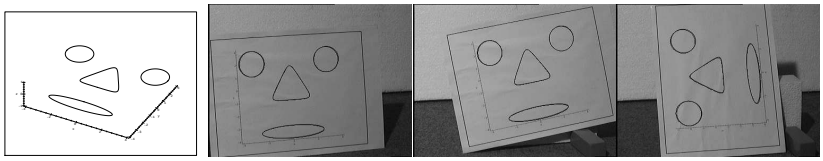

FIGURE 37.11. Pose estimation of an object containing one ellipse, two circles and one deltoid.

comparison to using the interpolated data. To visualize the pose quality the 3D object model is transformed and projected into the image. Figure 37.9 shows a more extreme case of disturbed image data. It can be seen, that the use of interpolated image data leads to more stable results than using the raw contour points.

Figure 37.10 shows the convergence behavior of our algorithm during the iterations. Since the rigid body motion to estimate is very large, the algorithm needs several iterations to converge. If only small movements are observed, the number of iterations (and therefore the computing time) can be reduced significantly.

Figure 37.11 shows pose estimation results of a second 3D object model. This object contains two circles, one ellipse and one deltoid. The left image shows the 3D object model. The other images show the transformed projected object model to visualize the quality of the pose.

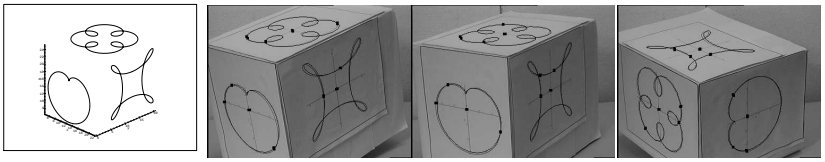

FIGURE 37.12. Pose estimation of an object containing a cardioid and two cycloids.

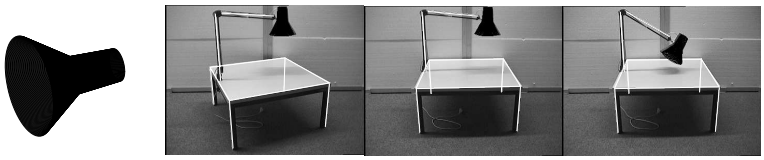

FIGURE 37.13. Pose estimation of 2twist surface, connected via a 2 d.o.f. kinematic chain to a table.

37.4.3 Pose estimation of non-convex objects

Now we look at pose estimation of non-convex objects. Figure 37.12 shows experimental results for an object, containing three cycloidal non-convex curves. We use all information simultaneously to solve for the pose parameters. Here we have to encapsulate the gradient method for pose estimation within a heuristic, since the entities are not convex any more. In the first image, the object model used is shown. The other three images show pose estimation results for the object. Again we mark the extracted point features in the image and the transformed projected points. Furthermore, the transformed projected cycloidal curves are shown. Since we measured the size of the object model by hand, the pose estimation result is reasonably accurate.

The combination of the gradient method with the heuristic mentioned leads to slow algorithms. While the pose for convex object models can be estimated in 10 frames per second, the estimation of the object presented in Figure 37.12 takes up to 5 minutes to converge to a global minimum. But indeed, it is possible to estimate the global minimum.

Figure 37.13 shows experimental results from a 2twist surface. The object model is a lamp-shade connected via a 2 d.o.f. kinematic chain to a table. In this experiment, the pose parameters and the angles of the kinematic chain are estimated. The lamp-shade itself is modeled by two conical parts. Since both surface parts are convex, the gradient method converges directly and we do not have to apply a heuristic to estimate the pose.

37.5 Discussion

In this contribution we present a new approach to estimate the pose of 2twist and 3twist curves. These cycloidal curves extend the classical pose estimation problems in an interesting and useful way. We believe that for many applications cycloidal curves are better suited to represent natural objects than the often used tessellated surfaces. The representation of these entities is very compact and geometric manipulations, like rigid body motions of the curves, can easily be performed, just by transforming the generating elements of the curve.

The main difficulties that occur in the context of 2D-3D pose estimation of general space curves or surfaces are the local minima in the error functions to be minimized. This occurs naturally in the case of non-convex curves. If the objects are convex (e.g., Figure 37.11) on the other hand, the gradient method for solving the pose parameters converges to the global minimum and the pose can be estimated in (nearly) video real-time (10 fps) on a SUN Ultra 10. But if the curves are non-convex, we have to deal with local minima within our error function. In this case we combine the previously used gradient method with a heuristic and then we are able to estimate the global minimum. But so far, the algorithm is too slow for practical applications. One possibility to deal with this problem is to separate the non-convex curve in a set of curves containing the convex parts. This was done in the experiment of Figure 37.13.

There the pose problem reduces to the simple case of convex object features, but it requires more a priori knowledge about the scenario. Future research will concentrate on faster algorithms to solve such constraint equations for non-convex curves. Furthermore, we will concentrate on free-form contours and surfaces. First results are presented in [17].

37.5.1 Acknowledgments

We would like to thank Christian Perwass and Jon Selig for the fruitful discussions and hints performing this work. Figures 37.5 and 37.6 are generated with the CLU draw programming package [3]. This work has been supported by DFG Graduiertenkolleg No. 357 and by EC Grant IST-2001-3422 (VISATEC).

37.6 REFERENCES

[1] Bayro-Corrochano, E., The geometry and algebra of kinematics. In [18], 2001, 457–472.

[2] Beveridge, J.R., Local search algorithms for geometric object recognition: Optimal correspondence and pose. *Ph.D. Thesis, Computer Science Department, University of Massachusetts, Amherst*, 1993. Available as Technical Report CS 93–5.

[3] CLU Library, A C++ Library for Clifford Algebra. Available at http://www.perwass.de/CLU *Lehrstuhl für kognitive Systeme, University Kiel*, 2001.

[4] O'Connor, J.J., and Robertson, E.F., Famous Curves Index http://www-history.mcs.st-andrews.ac.uk/history/Curves/Curves.html

[5] Gallier, J., *Geometric Methods and Applications. For Computer Science and Engineering.* Springer Verlag, New York Inc. 2001

[6] Grimson, W. E. L., *Object Recognition by Computer*. The MIT Press, Cambridge, MA, 1990.

[7] Hestenes, D., Li, H., and Rockwood, A., New algebraic tools for classical geometry. In [18], 2001, pp. 3–23.

[8] Hestenes, D., and Ziegler, R., Projective geometry with Clifford algebra. *Acta Applicandae Mathematicae*, Vol. **23**, 1991, 25–63.

[9] Horaud, R., Phong, T.Q., and Tao, P.D., Object pose from 2-d to 3-d point and line correspondences. *International Journal of Computer Vision*, Vol. **15**, 1995, 225–243.

[10] Li, H., Hestenes, D., and Rockwood, A., Generalized homogeneous coordinates for computational geometry. In [18], 2001, pp. 27–52.

[11] Lee, X., A Visual Dictionary of Special Plane Curves. http://xahlee.org/SpecialPlaneCurves_dir/specialPlaneCurves.html

[12] Murray, R.M., Li, Z., and Sastry, S.S., *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[13] Needham, T., *Visual Complex Analysis*. Oxford University Press, 1997.

[14] Perwass, C., and Lasenby, L., A unified description of multiple view geometry. In [18], 2001, pp. 337–369.

[15] Rosenhahn, B., Zhang, Y., and Sommer, G., Pose estimation in the language of kinematics. In: *Second International Workshop, Algebraic Frames for the Perception-Action Cycle*, AFPAC 2000, Sommer G. and Zeevi Y.Y. (Eds.), LNCS 1888, Springer-Verlag, Heidelberg, 2000, pp. 284–293.

[16] Rosenhahn, B., Granert, O., and Sommer, G., Monocular pose estimation of kinematic chains. In *Applied Geometric Algebras for Computer Science and Engineering*, Birkhäuser Verlag, Dorst L., Doran C. and Lasenby J. (Eds.), pp. 373–383, 2001.

[17] Rosenhahn, B., Perwass, C., and Sommer, G., Pose Estimation of 3D Free-form Contours. Technical Report 0207, University Kiel, 2002.

[18] Sommer, G., editor. *Geometric Computing with Clifford Algebra*. Springer Verlag, 2001.

[19] Walker, M.W., and Shao, L., Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Understanding*, Vol. **54**, No. 3, 1991, 358–367.

[20] Zhang, Z., Parameter estimation techniques: a tutorial with application to conic fitting. *Image and Vision Computing*, Vol. **15**, 1997, 59–76.

Bodo Rosenhahn
Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40, 24098 Kiel
Germany
E-mail: bro@ks.informatik.uni-kiel.de

Gerald Sommer
Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40, 24098 Kiel
Germany
E-mail: gs@ks.informatik.uni-kiel.de

Submitted: August 1, 2002.